Salter Intellectual Property Law
Jim H. Salter
105 Thoreau Lane
Folsom, CA 95630
408-406-4855 (office)
jim@salteriplaw.com
www.salteriplaw.com

facsimile transmittal

To:	Examiner Ben C. Wang Art Unit: 2192	**Fax:**	571-270-2240
From:	Jim H. Salter	**Date:**	09/13/2010
Re:	Appl. Ser. No. 10/688,573	**Pages:**	15
CC:			

x Urgent ☐ For Review ☐ Please Comment ☐ Please Reply ☐ Please Recycle

Dear Examiner Wang,

Please find attached an Interview Request Form and a Proposed Amendment. Teleconference call details are also attached.

Thank you.

Best Regards,

Jim Salter
105 Thoreau Lane
Folsom, CA 95630
408-406-4855 (office)
jim@salteriplaw.com

PTOL-413A (10-07)
Approved for use through 04/30/2008. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Applicant Initiated Interview Request Form

Application No.: 10/688,573 First Named Applicant: Robert M. Zeidman
Examiner: Ben C. Wang Art Unit: 2192 Status of Application: Pending

Tentative Participants:
(1) Examiner Wang (2) Jim H. Salter

(3) Robert M. Zeidman (4)

Proposed Date of Interview: Wed. Sept. 15, 2010 **Proposed Time:** 3pm ET (AM/PM)

Type of Interview Requested:
(1) [X] Telephonic (2) [ ] Personal (3) [ ] Video Conference
Teleconference call details attached.

Exhibit To Be Shown or Demonstrated: [ ] YES [X] NO
If yes, provide brief description:

Issues To Be Discussed

Issues (Rej., Obj., etc)	Claims/ Fig. #s	Prior Art	Discussed	Agreed	Not Agreed
(1) Rejection	1, 15, 22, 29	Lehman, Mathur	[ ]	[ ]	[ ]
(2)			[ ]	[ ]	[ ]
(3)			[ ]	[ ]	[ ]
(4)			[ ]	[ ]	[ ]

[ ] Continuation Sheet Attached

Brief Description of Arguments to be Presented:
Applicant wishes to discuss the pending claims, the cited references, and a proposed amendment.

An interview was conducted on the above-identified application on _____.
NOTE: This form should be completed by applicant and submitted to the examiner in advance of the interview (see MPEP § 713.01).
This application will not be delayed from issue because of applicant's failure to submit a written record of this interview. Therefore, applicant is advised to file a statement of the substance of this interview (37 CFR 1.133(b)) as soon as possible.

/ /
Applicant/Applicant's Representative Signature

Examiner/SPE Signature

Jim H. Salter
Typed/Printed Name of Applicant or Representative

35,668
Registration Number, if applicable

Conference Call Details:

To join the teleconference, please call one of the following numbers and enter the Participant Passcode (including the hash or pound key) when prompted.

Access Toll Free From USA:	866-212-0875
Access Toll Free From Canada:	866-212-7554
Direct Toll From Anywhere:	978-964-0049
Participant Passcode:	700099#

S/N 10/688,573 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Robert M. Zeidman	Examiner: Ben C. Wang
Serial No.:	10/688,573	Group Art Unit: 2192
Filed:	October 20, 2003	Docket No.: Zeid-01
Title:	SOFTWARE TOOL FOR SYNTHESIZING A REAL-TIME OPERATING SYSTEM	

PROPOSED AMENDMENT AND RESPONSE UNDER 37 CFR § 1.116

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

This Proposed Amendment and Response is filed in response to the Office Action mailed on September 1, 2010.

IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) A method for developing a real-time operating system, comprising:

specifying a set of n tasks, task(1) through task(n), to be scheduled for execution;

specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

using a data processor to synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks, said synthesized source code being executable on a target system after compilation; and

synthesizing source code from commands embedded in source code to control execution of said set of t init-tasks, wherein synthesizing source code from commands embedded in source code includes generating new source code based on the commands embedded in source code.

2. (Cancelled)

3. (Previously Presented) The method of claim 1) including specifying f f-loop tasks, each having an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

4. (Previously Presented) The method of claim 1) including specifying p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

5. (Previously Presented) The method of claim 1) including specifying c call-tasks, c being less

than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

6. (Previously Presented) The method of claim 1) including specifying r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

7. (Previously Presented) The method of claim 1) where tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

Claims 8-14 (Canceled).

15. (Currently Amended) An apparatus for developing a real-time operating system comprising:

a computer;

a non-transitory computer readable medium in data communication with the computer, the computer readable medium including a software synthesis program stored thereon, which when executed by the computer causes the computer to specify a set of n tasks, task(1) through task(n), to be scheduled for execution; specify t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n; synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks, said synthesized source code being executable on a target system after compilation; and synthesize source code from commands embedded in source code to control execution of said set of t init-tasks, wherein synthesizing source code from commands

embedded in source code includes generating new source code based on the commands embedded in source code.

16. (Cancelled)

17. (Previously Presented) The apparatus of claim 15 being configured to specify f f-loop tasks, each having an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

18. (Previously Presented) The apparatus of claim 15 being configured to specify p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

19. (Previously Presented) The apparatus of claim 15 being configured to specify c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

20. (Previously Presented) The apparatus of claim 15 being configured to specify r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the preemptive-task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

21. (Previously Presented) The apparatus of claim 15 wherein tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which

being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

22. (Currently Amended) An apparatus for developing a real-time operating system comprising:

a computer;

a non-transitory computer readable medium in data communication with the computer, the computer readable medium including a software synthesis program stored thereon, the software synthesis program including:

means for specifying a set of n tasks, task(1) through task(n), to be scheduled for execution;

means for specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

means for synthesizing source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks, said synthesized source code being executable on a target system after compilation; and

means for synthesizing source code from commands embedded in source code to control execution of said set of t init-tasks, wherein the means for synthesizing source code from commands embedded in source code includes means for generating new source code based on the commands embedded in source code.

23. (Cancelled)

24. (Previously Presented) The apparatus of claim 22 including means for specifying f f-loop tasks, each having have an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

25. (Previously Presented) The apparatus of claim 22 including means for specifying p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than

or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

26. (Previously Presented) The apparatus of claim 22 including means for specifying c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

27. (Previously Presented) The apparatus of claim 22 including means for specifying r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the preemptive-task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

28. (Previously Presented) The apparatus of claim 22 wherein tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which are ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

29. (Currently Amended) A non-transitory machine-readable medium embodying instructions which, when executed by a machine, cause the machine to:

specify a set of n tasks, task(1) through task(n), to be scheduled for execution;

specify t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks, said synthesized source code being executable on a target system after compilation; and

synthesize source code from commands embedded in source code to control execution of said set of t init-tasks, wherein synthesizing source code from commands embedded in source code includes generating new source code based on the commands embedded in source code.

30. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify f f-loop tasks, each having an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

31. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

32. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

33. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

34. (Previously Presented) The machine-readable medium of claim 29 wherein tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks,

all of which being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

REMARKS

This Proposed Amendment and Response is filed in response to the Office Action mailed on September 1, 2010. Please consider the above-identified patent application in view of the amendments and remarks provided herein.

Claims 1, 15, 22, and 29 are amended herein, no claims are canceled, and no claims are newly added; as a result, claims 1, 3-7, 15, 17-22, and 24-34 are pending in this application.

§103 Rejections of the Claims

Claims 1, 3, 5, 7, 15, 17, 19, 21-22, 24, 26, 28-30, 32, and 34 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Lehman, et al. (U.S. Patent No. 4,796,179, hereinafter, "Lehman") in view of Mathur et al. (U.S. Patent No. 6,671,745, hereinafter Mathur).

Claims 4, 18, 25, and 31 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Lehman, in view of Mathur, and further in view of Xu et al., "On Satisfying Timing Constraints in Hard-Real-Time Systems", 1991, ACM (hereinafter 'Xu').

Claims 6, 20, 27, and 33 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Lehman, in view of Mathur, and further in view of Yodaiken (U.S. Patent No. 5,995,745) (hereinafter 'Yodaiken').

Applicant respectfully submits that the Office Action did not make out a *prima facie* case of obviousness, because several of the cited references should be withdrawn from consideration as not proper prior art, and even if combined, the properly cited references fail to teach or suggest all of the claim elements of the pending claims of the present application.

The current Office Action and prior Office Actions admit that Lehman does not disclose most of the elements of amended claims 1, 15, 22, and 29. For example, the Office Action admits at pages 4-5 that the following elements of claim 1 are not taught by Lehman:

> specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;
>
> using a data processor to synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks, said synthesized source code being executable on a target system after compilation; and

synthesizing source code from commands embedded in source code to control execution of said set of t init-tasks.

Claims 15, 22, and 29 also include similar elements not taught or suggested by Lehman. As such, claims 1, 15, 22, and 29, and claims dependent thereon, are patentable over Lehman.

The Office Action offered Mathur as allegedly disclosing the elements listed above, which are missing from Lehman. Mathur describes a set of Application Program Interfaces (APIs) for a resource-limited environment. The APIs provide a mechanism for a computer application to interface with various components and modules of an operating system for a resource-limited environment. The APIs further provide a mechanism to interface with input/output devices commonly found in embedded systems running in a resource-limited environment. However, Mathur does not describe or suggest the source code synthesis recited in the present claims. In particular, exposing an API to an application and executing a call to an API as described in Mathur is not the same as using a data processor to synthesize source code from commands embedded in source code to implement the task scheduler or control execution of a set of t init-tasks, wherein synthesizing source code from commands embedded in source code includes generating new source code based on the commands embedded in source code. Mathur does not describe or suggest generating new source code based on the commands embedded in source code. As such, Mathur does not describe or suggest the embodiments as claimed.

The Office Action rejected claims 4, 18, 25, and 31 in part in view of Xu. The Office Action offered Xu as allegedly disclosing a means for specifying p-loop tasks. However, Xu does not disclose or suggest the elements missing from Lehman and Mathur as explained above. Therefore, Xu in combination with Lehman does not render the pending claims unpatentable.

Similarly, Yodaiken does not disclose or suggest the elements missing from Lehman, Mathur, and Xu as explained above. Therefore, Yodaiken in combination with Lehman, Mathur, and/or Xu does not render the pending claims unpatentable.

The Applicant respectfully submits that the cited references do not render obvious

the claims as presented. Therefore, the Applicant respectfully requests withdrawal of the §103(a) rejections.

CONCLUSION

Applicant respectfully submits that the claims are in condition for allowance, and notification to that effect is earnestly requested. The Examiner is invited to telephone Applicant's attorney, Jim H. Salter at 408-406-4855 to facilitate prosecution of this application.

Respectfully submitted,

ROBERT M. ZEIDMAN

By his Representatives,

Salter Intellectual Property Law
105 Thoreau Lane
Folsom, CA 95630
408-406-4855

Date September 13, 2010

By /jim h salter/
Jim H. Salter
Reg. No. 35,668

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being sent by facsimile at 571-270-2240 and addressed to: Attention: Mail Stop AF, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 13th day of September, 2010.

Jim H. Salter — /jim h salter/

Name — Signature